

**实 验 报 告**

**（ 2020 / 2021 学年 第一学期）**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 课程名称 | 单片机原理及应用 | | | | | |
| 实验名称 | 小型定时开关控制系统设计 | | | | | |
| 实验时间 | 2020 | 年 | 12 | 月 | 16 | 日 |
| 指导单位 | 计算机学院、软件学院、网络空间安全学院 | | | | | |
| 指导教师 | 倪晓军 | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 学生姓名 | 吴雯 | 班级学号 | B18030322 |
| 学院(系) | 计算机学院、软件学院、网络空间安全学院 | 专 业 | 计算机科学与技术 |

**实 验 报 告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验名称** | 小型定时开关控制系统设计 | | | **指导教师** | 倪晓军 |
| **实验类型** | 课内实验 | **实验学时** | 2 | **实验时间** | 2020-12-16 |

1. **实验目的和要求**
2. 掌握行列式键盘、LED、数码管、蜂鸣器、继电器等人机接口和机电设备的工作原理，以及使用单片机C语言对其进行控制的方法；
3. 掌握基于状态转移及定时调度的系统分析方法，并使用此方法对系统软件结构进行分析和设计，实现所要求的功能；
4. 掌握使用集成开发环境Keil进行单片机程序的设计、开发及调试的方法和过程。
5. **实验要求**
6. 通过单片机的IO端口控制人机接口及机电设备，完成一个定时开关的设计；
7. 定时开关的工作方式可设置为定时开或定时关；
8. 系统通过行列式键盘接受用户的按键输入，设置工作方式和定时时长；
9. 系统通过控制LED、数码管及蜂鸣器对用户的操作提供反馈和提示；
10. 当用户控制计时启动时，系统对用户设定的时长进行倒计时；
11. 如用户设置系统工作在定时开方式，则倒计时结束（计数到0）时控制继电器吸合；
12. 如用户设置系统工作在定时关方式，则倒计时开始时继电器吸合，倒计时结束（计数到0）时继电器断开。
13. **实验环境(实验设备)**

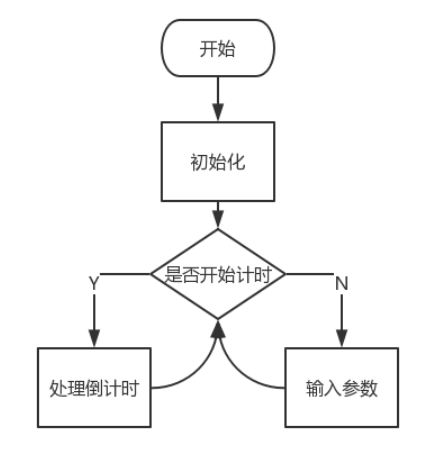
PC机、Keil C51集成开发环境、LBD简化版单片机教学实验系统。

1. **实验原理及内容**

系统的分析和设计：

1. 系统的分析：

系统核心为定时功能，在规定的时长后完成要求所需的功能动作，因此可采用系统内部的1号定时/计数器中断来实现。系统共存在两种状态：设置状态和倒计时状态， 可以通过是否开始计时来区分，因此主函数在初始化完成后主要完成两种状态的区分。在计时中断函数中，主要完成显示，输入和计时位的刷新工作。系统的简化时序流程 图如下：



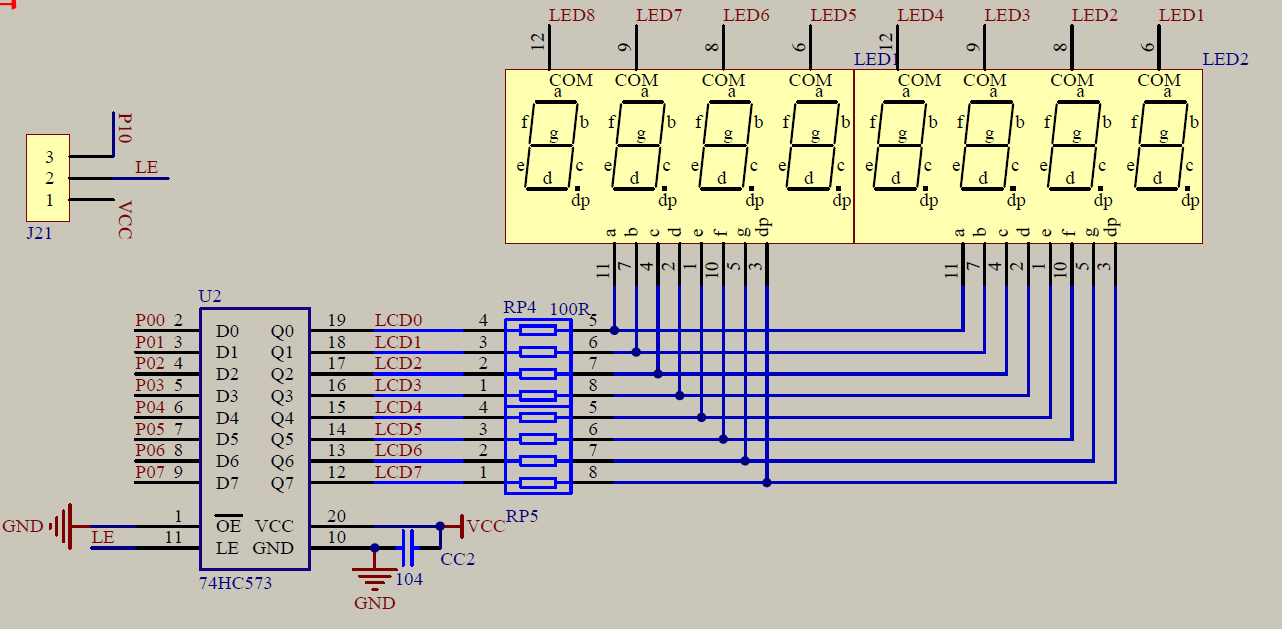
图一 系统简化时序流程图

1. 系统的设计：

系统所需的功能性模块包括：4\*4矩阵键盘用于设置倒计时时长，切换工作方式和开始倒计时；8位数码管用于显示剩余时间；继电器，用以完成规定动作；LED和蜂鸣器，用于反馈系统各个细节的运行情况。

定时/计数器，设置工作在方式一：16位定时器。在系统时钟频率为11.0582MHz的情况下，初值设置为0xDC00，每10ms触发一次定时/计数器中断。定时器计数器中断函数中，处理两件事：基于是否开始定时判断的时间刷新和键盘、数码管的刷新。具体为：每20ms，刷新一次键值和数码管；每1s判断后刷新计时位。

八位数码管，选用八位共阴极数码管，采用动态显示，为了节省I/O口，在数码管的段选位前接一个74HC573，用51P2口的2，3，4管脚作为数码管的位选信号。



图二 数码管接线原理图

4\*4矩阵键盘，采用行列刷新的方式获取按键值。由左至右，由上至下键值分别为0-15。其中键值对应功能如下表：

表一 键盘键值及功能



I/O口的接线的具体含义见下表：

表二 单片机I/O连接模块



关键代码的说明：

（1）定时/计数器的中断函数

定时器工作在方式一，其中频率为11.0592MHz。每1s判断一次是否修改计时位，每20ms刷新一次键值和数码管。

1. voidTimer() interrupt 1
2. {
3. TH0 = 0xDC;
4. TL0 = 0x00;
5. Count10ms++;
6. Count1s++;
7. **if**(Count1s == 100)
8. {
9. **if**(StartFlag == 1)
10. {
11. **if**(TimeArr[0] == 0)
12. {
13. **if**(TimeArr[1])
14. {
15. TimeArr[1]--;
16. TimeArr[0]= 60;
17. }
18. **else**
19. {
20. **if**(TimeArr[2])
21. {
22. TimeArr[2]--;
23. TimeArr[1]= 59;
24. TimeArr[0]= 60;
25. }
26. **else**
27. {
28. StartFlag= 0;
29. }
30. }
31. }
32. TimeArr[0]--;
33. }
34. Count1s = 0;
35. }
36. **if**(Count10ms == 2)
37. {
38. KeyDown();
39. DigDisplayTime();
40. Count10ms = 0;
41. }
42. }

（2）main函数

依照图一，实现的main函数，其中StartFlag为开始停止标志位。代码如下：

1. **void** main()
2. {
3. TimerInit();
4. ResetInit();
5. **while**(1)
6. {
7. RelayLed=~ ModLed;
8. **if**(StartFlag)
9. CountDown();
10. **else**
11. SetTime();
12. }
13. }

（3）数码管的显示函数

其中，LAS、LSB、LSC 为数码管段的输入信号。value为对应数码管显示数组smgduan的索引

1. **void** DigDisplayTime()
2. {
3. **int** i,value;
4. **for**(i=0;i<8;i++)
5. {
6. **switch**(i)
7. {
8. **case**(0):
9. LSA=0;LSB=0;LSC=0;value=TimeArr[0]%10;**break**;
10. **case**(1):
11. LSA=1;LSB=0;LSC=0;value=TimeArr[0]/10;**break**;
12. **case**(2):
13. LSA=0;LSB=1;LSC=0;value=11;**break**;
14. **case**(3):
15. LSA=1;LSB=1;LSC=0;value=TimeArr[1]%10;**break**;
16. **case**(4):
17. LSA=0;LSB=0;LSC=1;value=TimeArr[1]/10;**break**;
18. **case**(5):
19. LSA=1;LSB=0;LSC=1;value=11;**break**;
20. **case**(6):
21. LSA=0;LSB=1;LSC=1;value=TimeArr[2]%10;**break**;
22. **case**(7):
23. LSA=1;LSB=1;LSC=1;value=TimeArr[2]/10;**break**;
24. }
25. P0=smgduan[value];
26. delay(50);
27. P0=0x00;
28. }
29. }

（4）设置时间函数

根据KeyValue进行选择，每次选择后KeyValue归0。且在位移和加减选择中使用abs求绝对值和取模运算，避免出现负值乱码和超出范围的问题

1. **void** SetTime()
2. {
3. **switch**(KeyValue
4. {
5. **case**(StartStopKey):StartFlag = 1;**break**;
6. **case**(LeftSwitch):  SetTimeBit=(SetTimeBit+1)%3;**break**;
7. **case**(RightSwitch):    SetTimeBit=abs((SetTimeBit-1))%3;**break**;
8. **case**(PlusKey):        TimeArr[SetTimeBit]++;**break**;
9. **case**(SubKey):         TimeArr[SetTimeBit]=abs((TimeArr[SetTimeBit]-1))%100;**break**;
10. **case**(ModSwitch):      ModLed =~ ModLed;RelayLed=~ModLed;**break**;
11. **default**:**break**;
12. }
13. KeyValue = 0;
14. }

（5）倒计时函数

倒计时至最后5秒时，蜂鸣器发声作为继电器状态变化的提示

1. **void** CountDown()
2. {
3. u8 i;
4. **if**((TimeArr[1] ||TimeArr[2]) == 0 && TimeArr[0] < 5)
5. {
6. **for**(i=0;i<100;i++)
7. {
8. Beep=~ Beep;
9. delay(100);
10. }
11. }
12. **if**((TimeArr[0] ||TimeArr[1] || TimeArr[2]) == 0)
13. {
14. StartFlag =0;
15. RelayLed =~RelayLed;
16. ModLed =~ModLed;
17. }
18. }

（6）按键函数

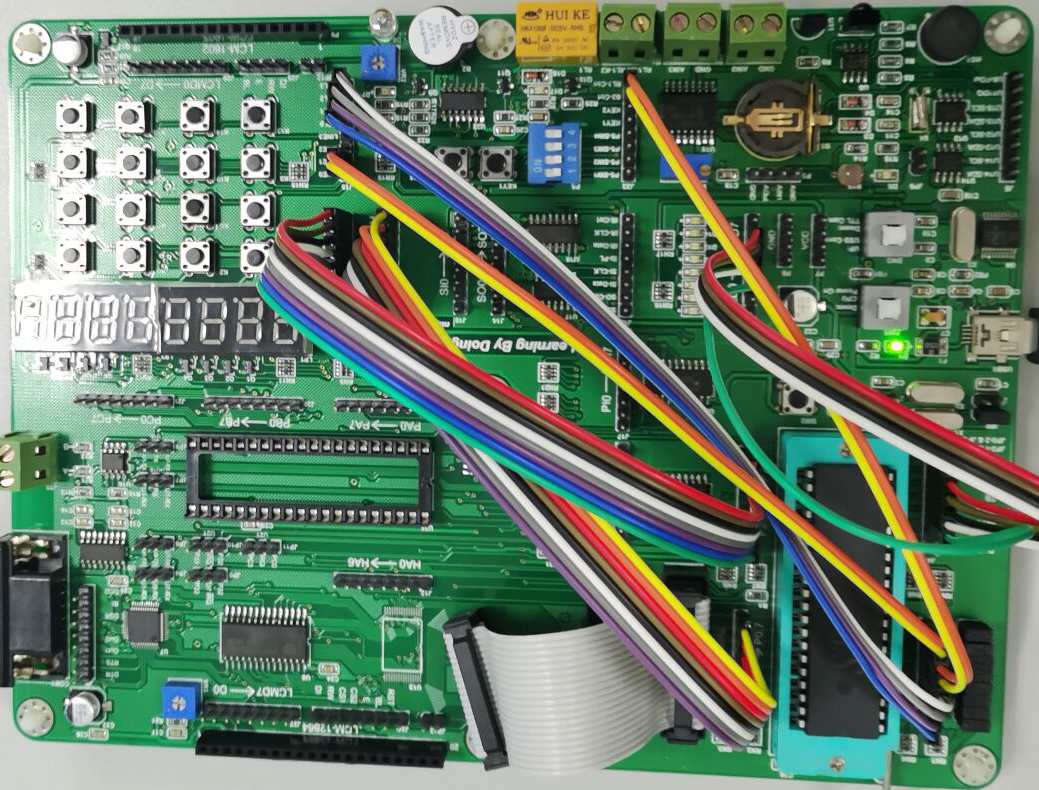
当有键按下时，蜂鸣器发声，作为反馈提示。

1. **void** KeyDown()
2. {
3. GPIO\_KEY=0x0f;
4. **if**(GPIO\_KEY!=0x0f)
5. {
6. Beep =~Beep;
7. delay(1000);
8. Beep =~Beep;
9. **if**(GPIO\_KEY!=0x0f)
10. {
11. GPIO\_KEY=0X0F;
12. **switch**(GPIO\_KEY)
13. {
14. **case**(0X07):      KeyValue=0;**break**;
15. **case**(0X0b):      KeyValue=1;**break**;
16. **case**(0X0d):KeyValue=2;**break**;
17. **case**(0X0e):      KeyValue=3;**break**;
18. }
19. GPIO\_KEY=0XF0;
20. **switch**(GPIO\_KEY)
21. {
22. **case**(0X70):      KeyValue=KeyValue;**break**;
23. **case**(0Xb0):      KeyValue=KeyValue+4;**break**;
24. **case**(0Xd0):KeyValue=KeyValue+8;**break**;
25. **case**(0Xe0):      KeyValue=KeyValue+12;**break**;
26. }
27. **while**(GPIO\_KEY!=0xf0);
28. }
29. }
30. **else**
31. KeyValue =0;
32. }

实验结果的分析：

设计使用电脑和单片机同时开始倒计时5小时，观察定时器误差。最终，当电脑时间归零时，定时器结果为00-12-14，误差为4%左右。

根据要求，系统实现良好。



1. **实验小结（包括问题和解决方法、心得体会、意见与建议等）**

通过对单片机小型定时开关控制系统设计，大致体会到单片机程序设计的流程。深刻体会到单片机主程序和中断程序的时序关系。在中断函数没有刷新标识位之前，主函数应当保持稳定。

实验过程中，起初使用一个int型值，来记录时间，结果在验证中发现，计时器在3小时27分左右会出现乱码。分析其原因，是由于实际值超出int型范围。后改为数组存放。

在按键中，发现按键后会出现乱码，分析后得出，由于计时位出现负值的缘故，因此在函数中增加取绝对值运算。

在按键中，还曾出现，多次位移后，加减无反应，程序奔溃的情况。分析后得出，为左右选择位超出计时位，导致数组索引溢出。在函数中增加了%取模运算，控制位移始终在数组索引范围内。

实现中，还出现过继电器和工作方式LED的交替高频闪烁问题。分析后得出，在每次按键后，没有将KeyValue归零，导致选择函数响应多次。

实验结果分析中，倒计时误差达4%。思考后可以通过如下几个方法降低计时误差：使用更高频的时钟信号，优化定时/计数器的中断函数，在中断函数中尽可能少的处理运算，通过标识位来传出中断必要的信息，优化显示函数，减少显示函数中的运算步骤。当然，这也存在问题，如果标识位变化时间小于每次变化的处理时间，则会造成处理中刷新，导致程序产生不可预期的结果

1. **指导教师评语**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **成 绩** |  | **批阅人** |  | **日 期** |  |